Express Mail Mailing Label No. EL 901 834 629 US
Date of Deposit August 24, 2001

Patent
Attorney's Docket No. 032481-034

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of	)	
	)	
Mukesh K. PATEL, et al.	)	Examiner: [prior application: C. Das]
	)	
Application No.: to be assigned	)	
	)	Group Art Unit: [prior application: 2122]
Filed: concurrently herewith	)	
	)	
For: JAVA VIRTUAL MACHINE HARDWARE FOR RISC AND CISC PROCESSORS	)	

PRELIMINARY AMENDMENT

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Prior to examination of the application wherein a request for continuation application is concurrently filed, please amend the application as follows:

In the Claims:

Please cancel claims 1-36.

Please add new claims 37-237 as follows.

37. A system comprising:

a central processing unit having a register file, the central processing unit adapted to execute register-based instructions; and

a hardware unit associated with the central processing unit, the hardware unit adapted to convert Java bytecode instructions into register-based instructions, wherein the hardware unit is adapted to store at least one Java variable in the central processing unit's register file at a location separate from any operand stack, wherein at least one of the register-based instructions reference a register in the central processing unit's register file containing one of the at least one Java variable.

38. The system of claim 37, wherein the central processing unit includes the hardware unit.

39. The system of claim 37, wherein the hardware unit is outside of the central processing unit.

40. The system of claim 37, wherein a portion of the operand stack is stored in the register file of the central processing unit and wherein the hardware unit is adapted to produce at least one of overflow or underflow indications for the portion of the operand stack stored in the register file.

41. The system of claim 37, wherein the hardware unit is further adapted to store at least some Java registers in the register file.

42. The system of claim 37, wherein the hardware unit implements at least part of a Java virtual machine.

43. The system of claim 37, wherein the hardware unit is connected between a memory and the central processing unit.

44. The system of claim 37, wherein the hardware unit is adapted to examine the stack-based instructions to determine whether multiple stack-based instructions can be combined into fewer register-based instructions.

45. The system of claim 44, wherein the hardware unit produces register-based instructions that access the Java variables in the register file so as to reduce the number of register-based instructions that would otherwise be required.

46. The system of claim 44, wherein multiple stack-based instructions pass through the hardware unit concurrently to allow for the operation of the combining logic.

47. The system of claim 44, wherein the hardware unit is adapted to convert multiple Java bytecodes into a single register-based instruction.

48. The system of claim 37, wherein the hardware unit produces an exception upon at least one of the stack-based instructions, and wherein the central processing unit will, in software, translate the at least one of the stack-based instructions causing the exception.

49. The system of claim 37, wherein the hardware unit is adapted to swap Java variables in and out of the register file from a memory.

50. The system of claim 37, wherein the hardware unit includes logic that keeps track of Java Variables stored in the register file and when a Java bytecode to be translated references a Java Variable stored in a register of the register file, the hardware unit produces an indication of that register to be used in the translation process.

51. The system of claim 37, wherein the hardware unit keeps track of which registers in the register file contain Java variables, the meaning of the registers being able to change as a result of an executed instruction.

52. A system comprising:

a central processing unit having a register file, the central processing unit adapted to execute register-based instructions; and

a hardware unit associated with the central processing unit, the hardware unit adapted to convert Java bytecode instructions into register-based instructions, wherein the

hardware unit is adapted to store at least one Java register in the central processing unit's register file, wherein at least one of the register-based instructions reference a register in the central processing unit's register file containing one of the at least one Java register, the at least one Java Register including the Java Program Counter.

53. The system of claim 52, wherein the central processing unit includes the hardware unit.

54. The system of claim 52, wherein the hardware unit is outside of the central processing unit.

55. The system of claim 52, wherein the central processing unit includes the hardware unit.

56. The system of claim 52, wherein the hardware unit is outside of the central processing unit.

57. The system of claim 52, wherein a portion of the operand stack is stored in the register file of the central processing unit and wherein the hardware unit is adapted to produce at least one of overflow or underflow indications for the portion of the operand stack stored in the register file.

58. The system of claim 52, wherein the hardware unit is further adapted to store at least some Java variables in the register file.

59. The system of claim 52, wherein the hardware unit implements at least part of a Java virtual machine.

60. The system of claim 52, wherein the hardware unit is connected between a memory and the central processing unit.

61. The system of claim 52, wherein the hardware unit is adapted to examine the stack-based instructions to determine whether multiple stack-based instructions can be combined into fewer register-based instructions.

62. The system of claim 61, wherein the hardware unit produces register-based instructions that access the Java registers in the register file so as to reduce the number of register-based instructions that would otherwise be required.

63. The system of claim 61, wherein multiple stack-based instructions pass through the hardware unit concurrently to allow for the operation of the combining logic.

64. The system of claim 52, wherein the hardware unit is adapted to convert multiple Java bytecodes into a single register-based instruction.

65. The system of claim 52, wherein the hardware unit produces an exception upon at least one of the stack-based instructions, and wherein the central processing unit will, in software, translate the at least one of the stack-based instructions causing the exception.

66. The system of claim 52, wherein the hardware unit is adapted to swap Java registers in and out of the register file from a memory.

67. A central processing unit comprising:

an input adapted to receive Java bytecode instructions;

a register file adapted to be manipulated using register-based instructions; and

a hardware subunit adapted to convert Java instructions into register-based instructions, wherein the hardware subunit is adapted to store at least one Java variable in the register file at a location separate from any operand stack, wherein at least one of the register-based instructions reference a register in the central processing unit's register file containing one of the at least one Java variable.

68. A central processing unit comprising:

an input adapted to receive Java bytecode instructions;

a register file adapted to be manipulated using register-based instructions; and

a hardware subunit adapted to convert Java instructions into register-based instructions, wherein the hardware subunit is adapted to store at least one Java register in the register file, wherein at least one of the register-based instructions reference a register in the central processing unit's register file containing one of the at least one Java register, the at least one Java Register including the Java Program Counter.

69. A system comprising:

a first unit adapted to execute register-based instructions, the first unit having an associated register file; and

a hardware unit associated with the first unit, the hardware unit adapted to convert stack-based instructions into register-based instructions, the hardware unit adapted to store portions of the operand stack in the first unit's register file, wherein the hardware unit includes logic to keep a count of how many entries have been placed on the operand stack.

70. The system of claim 69, wherein the first unit and the hardware unit are part of a central processing unit.

71. The system of claim 69, wherein the first unit comprises a central processing unit and wherein the hardware unit is outside of the central processing unit.

72. The system of claim 69, wherein a portion of the operand stack is stored in the register file of the first unit and wherein the hardware unit is adapted to produce at least one of overflow or underflow indications for the portion of the operand stack stored in the register file.

73. The system of claim 69, wherein the hardware unit is further adapted to store at least some Java registers in the register file.

74. The system of claim 69, wherein the hardware unit implements at least part of a Java virtual machine.

75. The system of claim 69, wherein the hardware unit is connected between a memory and the first unit.

76. The system of claim 69, wherein the hardware unit is adapted to manage a Java stack.

77 . The system of claim 69, wherein the hardware unit is adapted to examine the stack-based instructions to determine whether multiple stack-based instructions can be combined into fewer register-based instructions.

78. The system of claim 77, wherein the hardware unit produces register-based instructions that access the portion of the operand stack in the register file so as to reduce the number of register-based instructions that would otherwise be required.

79. The system of claim 77, wherein multiple stack-based instructions pass through the hardware unit concurrently to allow for the operation of the combining logic.

80. The system of claim 69, wherein the hardware unit is adapted to convert multiple Java bytecodes into a single register-based instruction.

81. The system of claim 80, wherein the multiple Java bytecodes include a basic operand instruction and one or more stack manipulation instructions.

82. The system of claim 80, wherein the multiple Java bytecodes includes a load or store instruction.

83. The system of claim 69, wherein the hardware unit produces an exception upon at least one of the stack-based instructions, and wherein the central processing unit will, in software, translate the at least one of the stack-based instructions causing the exception.

84. The system of claim 69, wherein the hardware unit is adapted to swap portions of the operand stack in and out of the register file from a memory.

85. The system of claim 69, wherein the hardware unit includes an indication of the depth of the portion of operand stack stored in the register file.

86. The system of claim 69, wherein the hardware unit includes logic to keep a count of how many entries have been placed on the operand stack.

87. The system of claim 69, wherein the hardware unit includes logic that keeps track of portions of the Java operand stack stored in the first unit's register file and when a Java bytecode to be translated references an element of the Java operand stack stored in a register of the first unit's register file, the hardware unit produces an indication of that register to be used in the translation process.

88. The system of claim 69, wherein the hardware unit keeps track of a top of stack register location, wherein the top of stack register in the first unit's register file is not fixed and can change as a result of an executed instruction.

89. The system of claim 69, wherein a portion of the Java operand stack or Java variables are stored in the register file of the first unit, wherein the hardware unit keeps track of which registers in the first unit's register file contains portions of the Java operand stack or Java variables, the meaning of the registers being able to change as a result of an executed instruction.

90. The central processing unit of claim 69, wherein an overflow or underflow produces operand transfer between the register file in the central processing unit and memory.

91. The central processing unit of claim 69, wherein the registers of the register file of the central processing unit used to store the portion of operand stack is full of valid data.

92. The central processing unit of claim 69, wherein the at least one of the overflow or underflow indications is generated by a stack instruction pushing an operand or popping the operand from the operand stack.

93. A system comprising:

a first unit adapted to execute register-based instructions, the first unit having an associated register file; and

a hardware unit associated with the first unit, the hardware unit adapted to convert Java bytecodes into register-based instructions, wherein the hardware unit includes logic that keeps track of portions of the Java operand stack or Java Variables stored in the first unit's register file and when a Java bytecode to be translated references an element of the Java operand stack or Java Variable stored in a register of the first unit's register file, the hardware unit produces an indication of that register to be used in the translation process.

94. The system of claim 93, wherein the first unit and the hardware unit are part of a central processing unit.

95. The system of claim 93, wherein the first unit comprises a central processing unit and wherein hardware unit is outside of the central processing unit.

96. The system of claim 93, wherein a portion of the operand stack is stored in the register file of the first unit and wherein the hardware unit is adapted to produce at least one of overflow or underflow indications for the portion of the operand stack stored in the register file.

97. The system of claim 93, wherein the hardware unit is further adapted to store at least some Java registers in the register file.

98. The system of claim 93, wherein the hardware unit implements at least part of a Java virtual machine.

99. The system of claim 93, wherein the hardware unit is connected between a memory and the first unit.

100. The system of claim 93, wherein the hardware unit is adapted to examine the stack-based instructions to determine whether multiple stack-based instructions can be combined into fewer register-based instructions.

101. The system of claim 100, wherein the hardware unit produces register-based instructions that access the portion of the operand stack in the register file so as to reduce the number of register-based instructions that would otherwise be required.

102. The system of claim 100, wherein multiple stack-based instructions pass through the hardware unit concurrently to allow for the operation of the combining logic.

103. The system of claim 93, wherein the hardware unit is adapted to convert multiple Java bytecodes into a single register-based instruction.

104. The system of claim 103, wherein the multiple Java bytecodes include a basic operand instruction and one or more stack manipulation instructions.

105. The system of claim 103, wherein the multiple Java bytecodes includes a load or store instruction.

106. The system of claim 93, wherein the hardware unit produces an exception upon at least one of the stack-based instructions, and wherein the first unit will, in software, translate the at least one of the stack-based instructions causing the exception.

107. The system of claim 93, wherein the hardware unit is adapted to swap Java variables or portions of the operand stack in and out of the register file from a memory.

108. The system of claim 93, wherein the hardware unit includes an indication of the depth of the portion of operand stack stored in the register file.

109. The system of claim 93, wherein the hardware unit includes logic to keep a count of how many entries have been placed on the operand stack.

110. The system of claim 93, wherein the hardware unit includes logic that keeps track of Java Variables stored in the first unit's register file and when a Java bytecode to be translated references a Java Variable stored in a register of the first unit's register file, the hardware unit produces an indication of that register to be used in the translation process.

111. The system of claim 93, wherein the hardware unit includes logic that keeps track of portions of the Java operand stack stored in the first unit's register file and when a Java bytecode to be translated references an element of the Java operand stack stored in a register of the first unit's register file, the hardware unit produces an indication of that register to be used in the translation process.

112. The system of claim 93, wherein the hardware unit keeps track of a top of stack register location, wherein the top of stack register in the first unit's register file is not fixed and can change as a result of an executed instruction.

113. The system of claim 93, wherein the hardware unit keeps track of which registers in the first unit's register file contains portions of the Java operand stack or Java variables, the meaning of the registers being able to change as a result of an executed instruction.

114. A system comprising:

a first unit adapted to execute register-based instructions, the first unit having an associated register file; and

a hardware unit associated with the first unit, the hardware unit adapted to convert Java bytecodes into register-based instructions, wherein the hardware unit includes logic that keeps track of Java Variables stored in the first unit's register file and when a Java bytecode to be translated references a Java Variable stored in a register of the first unit's register file, the hardware unit produces an indication of that register to be used in the translation process.

115. The system of claim 114, wherein the first unit and the hardware unit are part of a central processing unit.

116. The system of claim 114, wherein the first unit comprises a central processing unit and wherein the hardware unit is outside of the central processing unit.

117. The system of claim 114, wherein the hardware unit is further adapted to store at least some Java registers in the register file.

118. The system of claim 114, wherein the hardware unit is adapted to swap Java variables in and out of the register file from a memory.

119. A system comprising:

a first unit adapted to execute register-based instructions, the first unit having an associated register file; and

a hardware unit associated with the first unit, the hardware unit adapted to convert Java bytecodes into register-based instructions, wherein the hardware unit includes logic that keeps track of portions of the Java operand stack stored in the first unit's register file and when a Java bytecode to be translated references an element of the Java operand stack stored in a register of the first unit's register file, the hardware unit produces an indication of that register to be used in the translation process.

120. The system of claim 119, wherein the first unit and the hardware unit comprise a central processing unit.

121. The system of claim 119, wherein the first unit comprises a central processing unit and wherein the hardware unit is outside of the central processing unit.

122. The system of claim 119, wherein a portion of the operand stack is stored in the register file of the central processing unit and wherein the hardware unit is adapted to produce at least one of overflow or underflow indications for the portion of the operand stack stored in the register file.

123. The system of claim 119, wherein the hardware unit is adapted to examine the stack-based instructions to determine whether multiple stack-based instructions can be combined into fewer register-based instructions.

124. The system of claim 119, wherein the hardware unit produces register-based instructions that access the portion of the operand stack in the register file so as to reduce the number of register-based instructions that would otherwise be required.

125. The system of claim 119, wherein multiple stack-based instructions pass through the hardware unit concurrently to allow for the operation of the combining logic.

126. The system of claim 119, wherein the hardware unit is adapted to convert multiple Java bytecodes into a single register-based instruction.

127. The system of claim 119, wherein the multiple Java bytecodes include a basic operand instruction and one or more stack manipulation instructions.

128. The system of claim 119, wherein the hardware unit includes an indication of the depth of the portion of operand stack stored in the register file.

129. The system of claim 119, wherein the hardware unit includes logic to keep a count of how many entries have been placed on the operand stack.

130. The system of claim 119, wherein the hardware unit includes logic that keeps track of portions of the Java operand stack stored in the first unit's register file and when a Java bytecode to be translated references an element of the Java operand stack stored in a register of the first unit's register file, the hardware unit produces an indication of that register to be used in the translation process.

131. The system of claim 119, wherein the hardware unit keeps track of a top of stack register location, wherein the top of stack register in the first unit's register file is not fixed and can change as a result of an executed instruction.

132. The system of claim 119, wherein a portion of the Java operand stack or Java variables are stored in the register file of the first unit, wherein the hardware unit

keeps track of which registers in the first unit's register file contains portions of the Java operand stack or Java variables, the meaning of the registers being able to change as a result of an executed instruction.

133. A system comprising:

a first unit adapted to execute register-based instructions, the first unit having an associated register file; and

a hardware unit associated with the first unit, the hardware unit adapted to convert Java bytecodes into register-based instructions, wherein a portion of the operand stack is stored in the register file of the first unit, wherein the hardware unit keeps track of a top of stack register location, wherein the top of stack register in the first unit's register file is not fixed and can change as a result of an executed instruction.

134. The system of claim 133, wherein the first unit and the hardware unit includes the central processing unit.

135. The system of claim 133, wherein the first unit comprises a central processing unit and wherein the hardware unit is outside of the central processing unit.

136. The system of claim 133, wherein the hardware unit is adapted to produce at least one of overflow or underflow indications for the portion of the operand stack stored in the register file.

137. The system of claim 133, wherein the hardware unit is adapted to examine the stack-based instructions to determine whether multiple stack-based instructions can be combined into fewer register-based instructions.

138. The system of claim 137, wherein the hardware unit produces register-based instructions that access the portion of the operand stack in the register file so as to reduce the number of register-based instructions that would otherwise be required.

139. The system of claim 137, wherein multiple stack-based instructions pass through the hardware unit concurrently to allow for the operation of the combining logic.

140. The system of claim 133, wherein the hardware unit is adapted to convert multiple Java bytecodes into a single register-based instruction.

141. The system of claim 133, wherein the hardware unit produces an exception upon at least one of the stack-based instructions, and wherein the central processing unit will, in software, translate the at least one of the stack-based instructions causing the exception.

142. The system of claim 133, wherein the hardware unit includes an indication of the depth of the portion of operand stack stored in the register file.

143. The system of claim 133, wherein the hardware unit includes logic to keep a count of how many entries have been placed on the operand stack.

144. The system of claim 133, wherein the hardware unit includes logic that keeps track of portions of the Java operand stack stored in the first unit's register file and when a Java bytecode to be translated references an element of the Java operand stack stored in a register of the first unit's register file, the hardware unit produces an indication of that register to be used in the translation process.

145. The system of claim 133, wherein the hardware unit keeps track of which registers in the first unit's register file contains portions of the Java operand stack or Java variables, the meaning of the registers being able to change as a result of an executed instruction.

146. A system comprising:

a first unit adapted to execute register-based instructions, the first unit having an associated register file; and

a hardware unit associated with the first unit, the hardware unit adapted to convert Java bytecodes into register-based instructions, wherein a portion of the Java operand stack or Java variables are stored in the register file of the first unit, wherein the hardware unit keeps track of which registers in the first unit's register file contains portions of the Java operand stack or Java variables, the meaning of the registers being able to change as a result of an executed instruction.

147. The system of claim 146, wherein a top of stack register in the first unit's register file is not fixed and can change as a result of an executed instruction.

148. The system of claim 146, wherein the first unit and the hardware unit comprise a central processing unit.

149. The system of claim 146, wherein the first unit comprises a central processing unit and the hardware unit is outside of the central processing unit.

150. The system of claim 146, wherein the hardware unit is adapted to produce at least one of overflow or underflow indications for the portion of the operand stack stored in the register file.

151. The system of claim 146, wherein the hardware unit is adapted to examine the stack-based instructions to determine whether multiple stack-based instructions can be combined into fewer register-based instructions.

152. The system of claim 151, wherein the hardware unit produces register-based instructions that access the portion of the operand stack in the register file so as to reduce the number of register-based instructions that would otherwise be required.

153. The system of claim 151, wherein multiple stack-based instructions pass through the hardware unit concurrently to allow for the operation of the combining logic.

154. The system of claim 146, wherein the hardware unit is adapted to convert multiple Java bytecodes into a single register-based instruction.

155. The system of claim 154, wherein the multiple Java bytecodes include a basic operand instruction and one or more stack manipulation instructions.

156. The system of claim 146, wherein the hardware unit includes an indication of the depth of the portion of operand stack stored in the register file.

157. The system of claim 146, wherein the hardware unit includes logic to keep a count of how many entries have been placed on the operand stack.

158. The system of claim 146, wherein the hardware unit includes logic that keeps track of portions of the Java operand stack stored in the first unit's register file and when a Java bytecode to be translated references an element of the Java operand stack stored in a

register of the first unit's register file, the hardware unit produces an indication of that register to be used in the translation process.

159. A system comprising:

a first unit adapted to execute register-based instructions, the first unit having at least two associated register files of registers, the first set of register files used for normal operation, the second set of register files used for operation in a hardware translation mode; and

a hardware unit associated with the first unit, the hardware unit adapted to convert stack-based instructions into register-based instructions during the hardware translation mode.

160. A system comprising:

a first unit adapted to execute register-based instructions, the first unit adapted to produce a "branch taken" indication; and

a hardware unit associated with the first unit, the hardware unit adapted to convert stack-based instructions into register-based instructions, the hardware unit containing multiple pipeline stages, the pipeline stages being flushed when the "branch taken" indication is produced.

161. A system comprising:

a first unit adapted to execute register-based instructions; and

a hardware unit associated with the first unit, the hardware unit adapted to convert stack-based instructions into register-based instructions, the hardware unit including a stack-based instruction buffer associated with a decode unit.

162. The system of claim 161, wherein decode unit can select multiple stack-based instructions to decode at one time.

163. A system comprising:

a first unit adapted to execute register-based instructions; and

a hardware unit associated with the first unit, the hardware unit adapted to convert stack-based instructions into register-based instructions, the hardware unit including a stack-based instruction buffer associated with a decode unit, the hardware unit adapted to rearrange the stack-based instructions in the buffer to make them easier to be operated on by a decode unit.

164. The system of claim 164 wherein decode unit can select multiple stack-based instructions to decode at one time.

165. The system of claim 164 wherein the any stack-based instruction or instructions sent to the decoder unit for translation are removed from the stack-based instruction buffer and new stack-based instruction or instructions added to the buffer.

166. A system comprising:

a first unit adapted to execute register-based instructions; and

a hardware unit associated with the first unit, the hardware unit adapted to convert stack-based instructions into register-based instructions, wherein certain instructions are not translated in the hardware unit but instead cause translation software to be loaded into the first unit.

167. A system comprising:

a central processing unit having a register file, the central processing unit adapted to execute register-based instructions; and

a hardware unit associated with the central processing unit, the hardware unit adapted to convert stack-based instructions into register-based instructions, wherein a portion of the operand stack is stored in the register file of the central processing unit and wherein the hardware unit is adapted to produce at least one of overflow or underflow indications for the portion of the operand stack stored in the register file.

168. The system of claim 167, wherein the central processing unit includes the hardware unit.

169. The system of claim 167, wherein the hardware unit is outside of the central processing unit.

170. The system of claim 167, wherein the hardware unit is adapted to swap parts of the operand stack in and out of the register file from a memory.

171. The system of claim 167, wherein the system includes an indication of the depth of the portion of operand stack stored in the register file.

172. The system of claim 167, wherein the indication of the operand stack depth is stored in the hardware unit.

173. The system of claim 167, wherein a overflow or underflow produces operand transfer between the register file in the central processing unit and memory.

174. The system of claim 167, wherein, the stack based instructions are Java bytecodes

175. The system of claim 167, wherein the registers of the register file of the central processing unit used to store the portion of operand stack is full of valid data.

176. The system of claim 175, wherein the at least one of the overflow or underflow indications is generated by a stack instruction pushing an operand or popping the operand from the operand stack.

177. The system of Claim 167, wherein the hardware unit has access to the data bus of the central processing unit.

178. The system of Claim 167, wherein the hardware unit is further adapted to store at least some Java variables in the register file.

179. The system of Claim 167, wherein the hardware unit is further adapted to store at least some Java registers in the register file.

180. The system of Claim 167, wherein the stack-based instructions are associated with a virtual machine.

181. The system of Claim 167, wherein the stack-based instructions are Java bytecode.

182. The system of Claim 167, wherein the hardware unit implements at least part of a Java virtual machine.

183. The system of Claim 167, wherein the hardware unit is connected between a memory and the central processing unit.

184. The system of Claim 183, wherein the hardware unit is connected between an instruction cache and the central processing unit.

185. The system of Claim 167, wherein the hardware unit is adapted to manage a Java stack.

186. The system of Claim 167, wherein the hardware unit has access to at least one bus of the central processing unit.

187. The system of Claim 167, wherein the hardware unit is adapted to examine the stack-based instructions to determine whether multiple stack-based instructions can be combined into fewer register-based instructions.

188. The system of Claim 187, wherein the hardware unit produces register-based instructions that access the portion of the operand stack in the register file so as to reduce the number of register-based instructions that would otherwise be required.

189. The system of Claim 187, wherein multiple stack-based instructions pass through the hardware unit concurrently to allow for the operation of the combining logic.

190. The system of Claim 167, wherein the hardware unit is adapted to convert multiple Java bytecodes into a single register-based instruction.

191. The system of Claim 190, wherein the multiple Java bytecodes include a basic operand instruction and one or more stack manipulation instructions.

192. The system of Claim 190, wherein the multiple Java bytecodes includes a load or store instruction.

193. The system of Claim 167, wherein the central processing unit and hardware unit are on the same chip.

194. The system of Claim 167, wherein the hardware unit produces an exception upon at least one of the stack-based instructions, and wherein the central processing unit will, in software, translate the at least one of the stack-based instructions causing the exception.

195. The system of claim 167, wherein the hardware unit includes logic to keep a count of how many entries have been placed on the operand stack.

196. The system of claim 167, wherein the hardware unit includes logic that keeps track of portions of the Java operand stack stored in the register file and when a Java bytecode to be translated references an element of the Java operand stack stored in a register of the register file, the hardware unit produces an indication of that register to be used in the translation process.

197. The system of claim 167, wherein the hardware unit keeps track of a top of stack register location, wherein the top of stack register in the register file is not fixed and can change as a result of an executed instruction.

198. The system of claim 167, wherein the hardware subunit keeps track of which registers in the register file contain portions of the Java operand stack, the meaning of the registers being able to change as a result of an executed instruction.

199. The system of claim 168, wherein the central processing unit includes an execution unit to execute the register-based instructions.

200. The system of claim 167, wherein the translated register-based instructions are produced internally within the central processing unit.

201. The system of claim 167, wherein register-based instructions cause the manipulation of the register file.

202. A central processing unit comprising:

an input adapted to receive stack-based instructions;

a register file adapted to be manipulated using register-based instructions, the register file adapted to store a portion of an operand stack; and

a hardware subunit adapted to convert stack-based instructions into register-based instructions, wherein the hardware subunit is adapted to produce at least one of overflow or underflow indications for the portion of the operand stack stored in the register file.

203. The central processing unit of claim 202, wherein, the stack-based instructions are Java bytecodes.

204. The central processing unit of Claim 202, wherein the hardware subunit is adapted to swap parts of the operand stack in and out of the register file from a memory.

205. The central processing unit of Claim 202, wherein the hardware subunit is further adapted to store at least some Java variables in the register file.

206. The central processing unit of Claim 202, wherein the hardware subunit is further adapted to store at least some Java registers in the register file.

207. The central processing unit of Claim 202, wherein the central processing unit includes an indication of the depth of the portion of operand stack stored in the register file.

208. The central processing unit of Claim 202 wherein the indication of the operand stack depth is stored in the hardware subunit.

209. The central processing unit of Claim 202 wherein a overflow or underflow produces operand transfer between the register file in the central processing unit and memory.

210. The central processing unit of Claim 202 wherein, the stack-based instructions are Java bytecodes.

211. The central processing unit of Claim 202, wherein the registers of the register file of the central processing unit used to store the portion of operand stack is full of valid data.

212. The central processing unit of Claim 202, wherein the at least one of the overflow or underflow indications is generated by a stack instruction pushing an operand or popping the operand from the operand stack.

213. The central processing unit of Claim 202, wherein the stack-based instructions are associated with a virtual machine.

214. The central processing unit of Claim 202, wherein the stack-based instructions are Java bytecodes.

215. The central processing unit of Claim 202, wherein the hardware subunit implements at least part of a Java virtual machine.

216. The central processing unit of Claim 202, wherein the hardware subunit is adapted to manage a Java stack.

217. The central processing unit of Claim 202, wherein the hardware subunit has access to at least one bus of the central processing unit.

218. The central processing unit of Claim 202, wherein the hardware subunit is adapted to examine the stack-based instructions to determine whether multiple stack-based instructions can be combined into fewer register-based instructions.

219. The central processing unit of Claim 218, wherein the hardware subunit produces register-based instructions that access the portion of the operand stack in the register file so as to reduce the number of register-based instructions that would otherwise be required.

220. The central processing unit of Claim 218, wherein multiple stack-based instructions pass through the hardware subunit concurrently to allow for the operation of the combining logic.

221. The central processing unit of Claim 202, wherein the hardware subunit is adapted to convert multiple Java bytecodes into a single register-based instruction.

222. The central processing unit of Claim 221, wherein the multiple Java bytecodes include a basic operand instruction and one or more stack manipulation instructions.

223. The central processing unit of Claim 221, wherein the multiple Java bytecodes includes a load or store instruction.

224. The central processing unit of Claim 202, wherein the hardware subunit produces an exception upon at least one of the stack-based instructions, and wherein the central processing unit will, in software, translate the at least one of the stack-based instructions causing the exception.

225. The system of claim 202, wherein the hardware subunit includes logic to keep a count of how many entries have been placed on the operand stack.

226. The system of claim 202, wherein the hardware subunit includes logic that keeps track of Java variables stored in the register file and when a Java bytecode to be translated references a Java variable stored in a register of the register file, the hardware subunit produces an indication of that register to be used in the translation process.

227. The system of claim 202, wherein the hardware subunit includes logic that keeps track of portions of the Java operand stack stored in the register file and when a Java bytecode to be translated references an element of the Java operand stack stored in a register of the register file, the hardware unit produces an indication of that register to be used in the translation process.

228. The system of claim 202, wherein the hardware subunit keeps track of which registers in the register file contain portions of the Java operand stack, the meaning of the registers being able to change as a result of an executed instruction.

229. The system of claim 202, wherein the central processing unit includes an execution unit to execute the register-based instructions.

230. The system of claim 202, wherein the translated register-based instructions are produced internally within the central processing unit.

231. A system comprising:

an execution unit associated with a register file, the execution unit adapted to execute decoded instructions; and

hardware adapted to receive Java bytecodes and native non-Java instructions and adapted to produce decoded instructions to the execution unit, the hardware including a java hardware unit adapted to store at least one Java variable in the register file at a location separate from any operand stack, wherein at least one of the decoded instructions reference a register in the central processing unit's register file containing one of the at least one Java Variable, wherein a portion of the operand stack is stored in the register file and wherein the hardware unit is adapted to produce at least one of overflow or underflow indications for the portion of the operand stack stored in the register file.

232. The system of claim 231, wherein the execution unit and associated register file are part of a central processing unit.

233. The system of claim 232, wherein the java hardware unit is part of the central processing unit.

234. The system of claim 232, wherein the java hardware unit is outside of the central processing unit.

235. The system of claim 234, wherein the java hardware unit is further adapted to translate Java bytecodes into native instructions.

236. The system of claim 235, wherein the hardware includes portions of the central processing unit, the portions including a decoder.

237. The system of claim 231, wherein the java hardware unit is further adapted to translate Java bytecodes into native instructions.

Respectfully submitted,

BURNS, DOANE, SWECKER & MATHIS, L.L.P.

By: _____

Joseph P. O'Malley
Registration No. 36,226
Redwood Shores, CA office
Tel: (650) 622-2333

P.O. Box 1404
Alexandria, Virginia 22313-1404

Date: August 23, 2001

21839

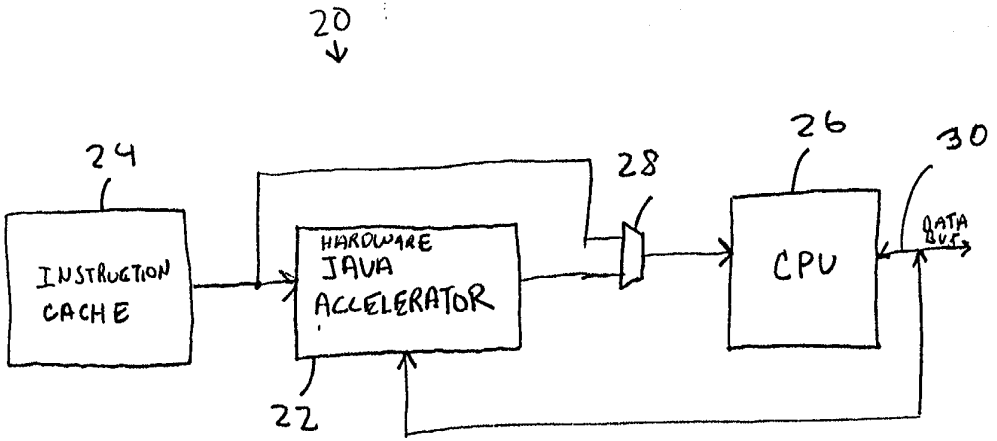
20
24
INSTRUCTION CACHE
HARDWARE JAVA ACCELERATOR
22
28
26
CPU
30
DATA BUS

FIGURE 1

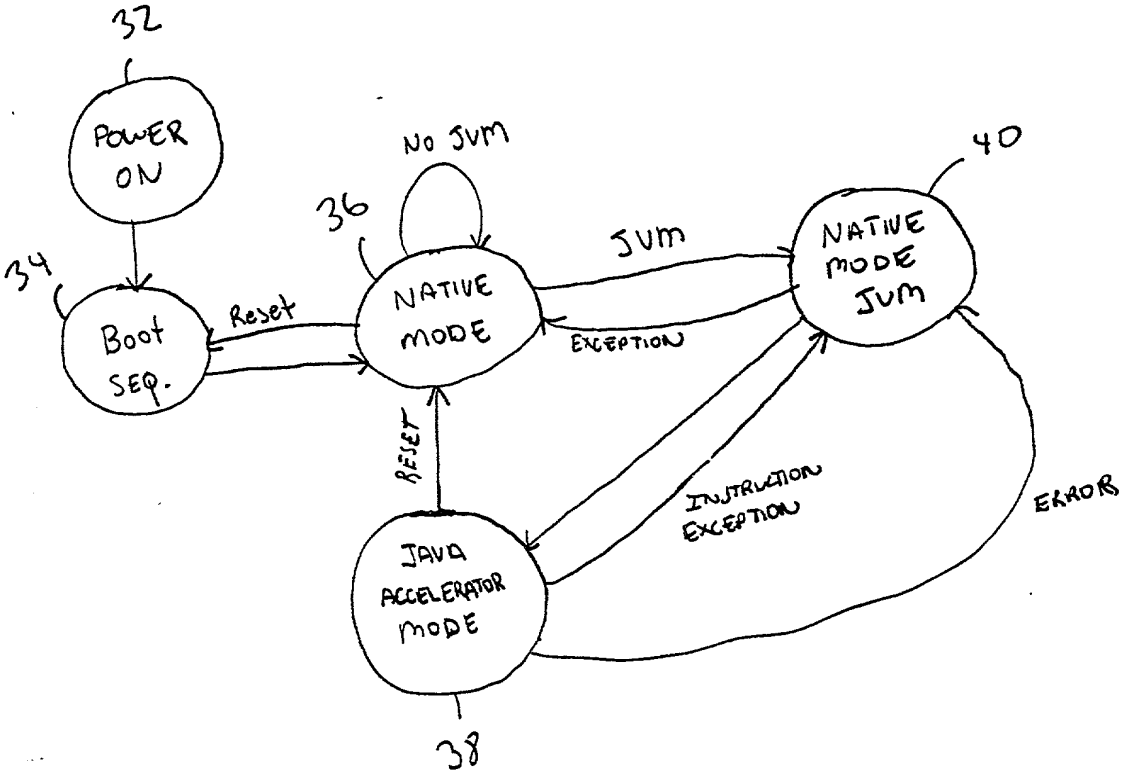
32
POWER ON
34
Boot SEQ.
Reset
36
NATIVE MODE
No JVM
JVM
EXCEPTION
40
NATIVE MODE JVM
RESET
INSTRUCTION EXCEPTION
ERROR
JAVA ACCELERATOR MODE
38

FIGURE 2

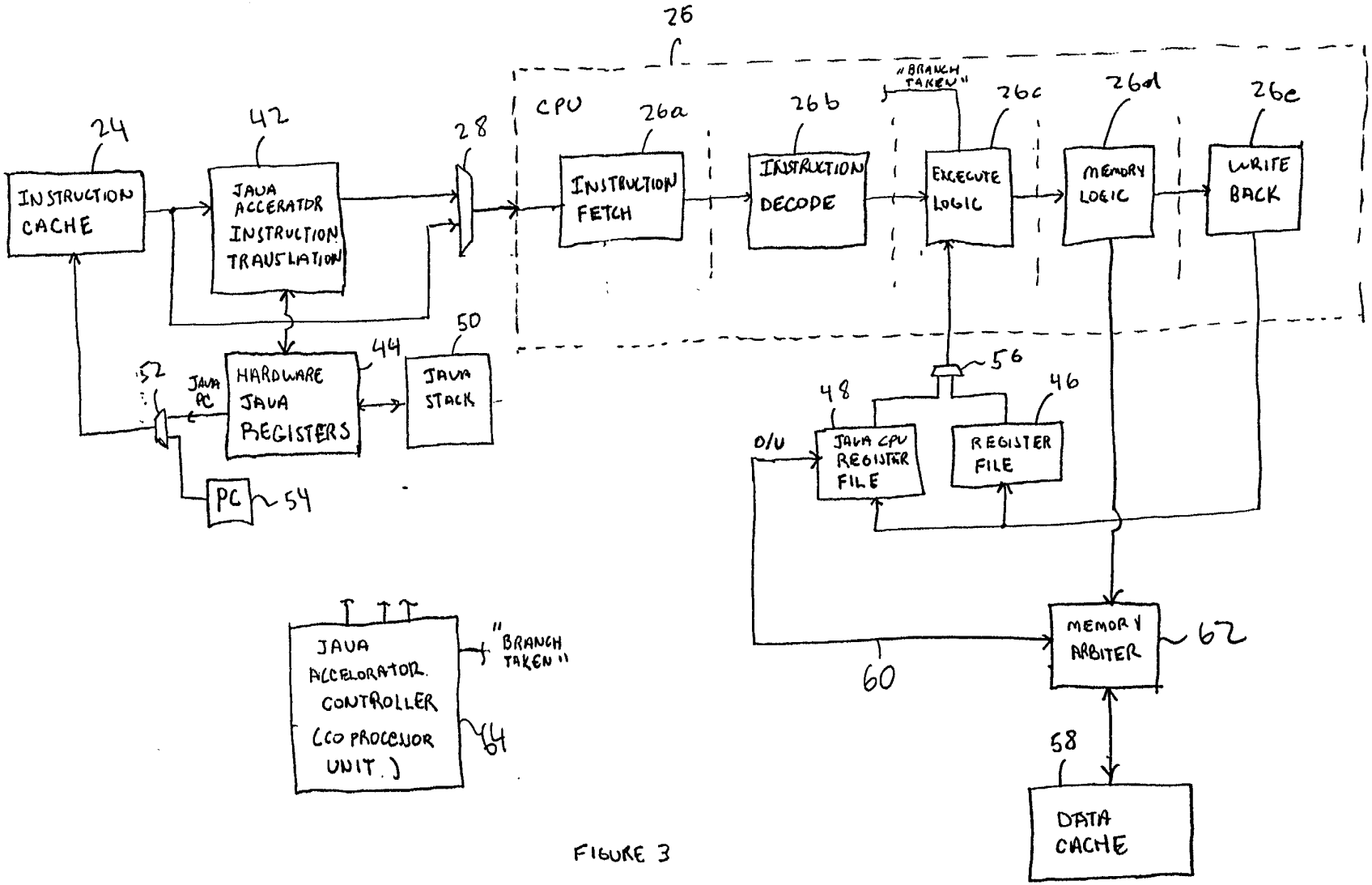
26
CPU
"BRANCH TAKEN"
24
42
28
26a
26b
26c
26d
26e
INSTRUCTION CACHE
JAVA ACCERATOR INSTRUCTION TRANSLATION
INSTRUCTION FETCH
INSTRUCTION DECODE
EXECUTE LOGIC
MEMORY LOGIC
WRITE BACK
50
44
52
JAVA PC
HARDWARE JAVA REGISTERS
JAVA STACK
56
48
46
D/U
JAVA CPU REGISTER FILE
REGISTER FILE
PC
54
JAVA ACCELORATOR CONTROLLER (CO PROCESSOR UNIT)
"BRANCH TAKEN"
64
60
MEMORY ARBITER
62
58
DATA CACHE

FIGURE 3

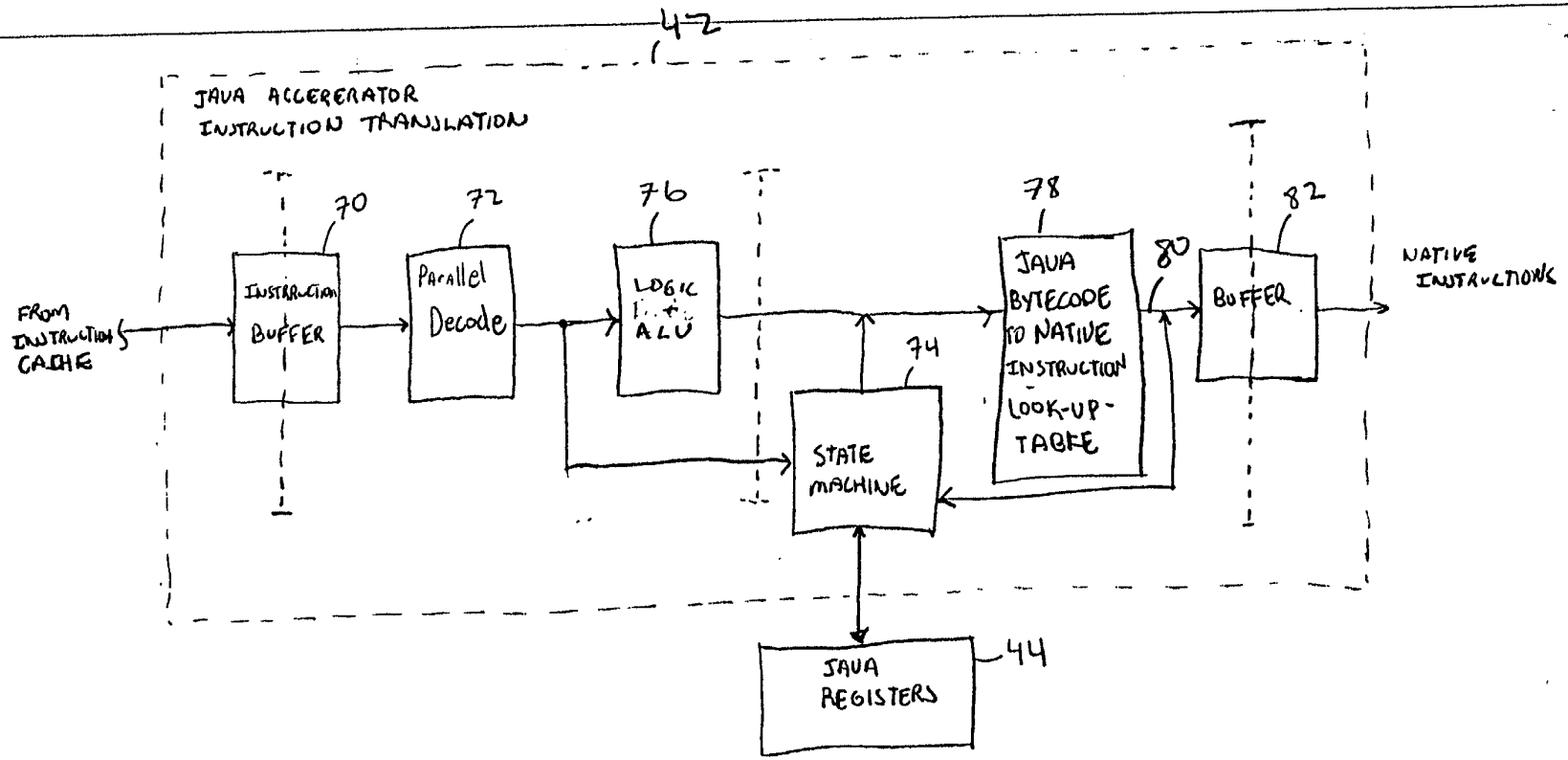

FIGURE 4

I. INSTRUCTION TRANSLATION

JAVA BYTECODE	⇒	NATIVE INSTRUCTION
iadd		ADD R1, R2

II. JAVA REGISTER

PC = VALUE A	⇒	PC = VALUE A + 1
OPTOP = VALUE B (R1)		OPTOP = VALUE B-1 (R2)
VAR = VALUE C		VAR = VALUE C

III. JAVA CPU REGISTER FILE

	R0	0001		R0	0001
contains VALUE of top of operand stack →	R1	0150	Not a valid stack value →	R1	0150
	R2	1210	contains →	R2	1360
	R3	0007	value of the top of operand stack	R3	0007
	R4	0005		R4	0005
	R5	0006		R5	0006
contains first VARIABLE →	R6	1221		R6	1221
	R7	1361		R7	1361

IV. MEMORY

OPTOP =	VALUE B ⇒	0150			0150
	(VALUE B-1)	1210	OPTOP = VALUE B-1		1360
		0007			0007
		0005			0005
		0006			0006
		0001			0001
		4427			4427
VAR =	VALUE C	1221	VAR = VALUE C		1221
		1361			1361
		1101			1101

FIGURE 5

I. INSTRUCTION TRANSLATION

JAVA BYTECODE		NATIVE INSTRUCTION
iload_n iadd	⇒	ADD R6, R1

II. JAVA REGISTER

PC = VALUE A		PC = VALUE A + 2
OPTOP = VALUE B (R1)	⇒	OPTOP = VALUE B (R1)
VAR = VALUE C		VAR = VALUE C

III. JAVA CPU REGISTER FILE

	R0	0001		R0	0001
contains VALUE OF TOP OF OPERAND STACK →	R1	0150	⇒ contains value of top of stack →	R1	1371
	R2	1210		R2	1210
	R3	0007		R3	0007
	R4	0005		R4	0005
	R5	0006		R5	0006
CONTAINS FIRST VARIABLE →	R6	1221	Contains first VARIABLE →	R6	1221
	R7	1361		R7	1361

IV. MEMORY

OPTOP = VALUE B	-	0150	OPTOP = VALUE B	-	1371
	-	1210		-	1210
	-	0007		-	0007
	-	0005		-	0005
	-	0006		-	0006
	-	0001		-	0001
	-	4427		-	4427
VAR = VALUE C	-	1221	VAR = VALUE C	-	1221
	-	1361		-	1361
	-	1101		-	1101

FIGURE 6

Opcodes Mnemonic	Opcode xHH	Excep Gen
nop	0x00	
aconst_null	x01	
iconst_m1	x02	
iconst_n(0-5)	x03 - x08	
lconst_n(0-1)	x09 - x0a	
fconst_n(0-2)	x0c - x0d	
dconst_n(0-1)	x0e -x0f	
bipush	x10	
sipush	x11	
ldc	x12	y
ldc_w	x13	y
ldc2_w	x14	y
iload	x15	
lload	x16	
fload	x17	
dload	x18	
aload	x19	
iload_n(0-3)	x1a - x1d	
lload_n(0-3)	x1e - x21	
fload_n(0-3)	x22 - x25	
dload_n(0-3)	x26 - x29	
aload_n(0-3)	x2a - x2d	
iaload	x2e	
laload	x2f	
faload	x30	
daload	x31	
aaload	x32	
baload	x33	
caload	x34	
saload	x35	
istore	x36	
lstore	x37	
fstore	x38	
dstroe	x39	
astroe	x3a	
istore_n(0-3)	x3b - x3e	
lstore_n(0-3)	x3f - x42	
fstore_n(0-3)	x43 - x46	
dstore_n(0-3)	x47 - x4a	
astore_n(0-3)	x4b - x4e	
iastore	x4f	
lastore	x50	
fastroe	x51	
dastore	x52	
bastore	x53	
aastore	x54	
castroe	x55	
sastore	x56	

FIGURE 7A

pop	x57	
pop2	x58	
dup	x59	
dup_x1	x5a	
dup_x2	x5b	
dup2	x5c	
dup2_x1	x5d	
dup2_x2	x5e	
swap	x5f	
iadd	x60	
ladd	x61	
fadd	x62	y
dadd	x63	y
isub	x64	
lsub	x65	
fsub	x66	y
dsub	x67	y
imul	x68	
lmul	x69	
fmul	x6a	y
dmul	x6b	y
idiv	x6c	y
ldiv	x6d	y
fdiv	x6e	y
ddiv	x6f	y
irem	x70	y
lrem	x71	y
frem	x72	y
drem	x73	y
ineg	x74	
lneg	x75	
fneg	x76	y
dneg	x77	y
ishl	x78	
lshl	x79	
ishr	x7a	
lshr	x7b	
iushr	x7c	
lushr	x7d	
iand	x7e	
land	x7f	
ior	x80	
lor	x81	
ixor	x82	
lxor	x83	
iinc	x84	
i2l	x85	y
i2f	x86	y
i2d	x87	y
l2i	x88	y
l2f	x89	y
l2d	x8a	y

FIGURE 7B

f2i	x8b	y
f2l	x8c	y
f2d	x8d	y
d2i	x8e	y
d2l	x8f	y
d2f	x90	y
i2b	x91	
i2c	x92	
i2s	x93	
lcmp	x94	y
fcmpl	x95	y
fcmpg	x96	y
dcmpl	x97	y
dcmpg	x98	y
ifeq	x99	
ifne	x9a	
iflt	x9b	
ifge	x9c	
ifgt	x9d	
ifle	x9e	
if_icmpeq	x9f	
if_icmpne	xa0	
if_icmplt	xa1	
if_acmpge	xa2	
if_cmpgt	xa3	
if_icmple	xa4	
if_acmpeq	xa5	
if_acmpne	xa6	
goto	xa7	
jsr	xa8	
ret	xa9	
tableswitch	xaa	y
lookupswitch	xab	y
ireturn	xac	
lreturn	xad	
freturn	xae	
dreturn	xaf	
areturn	xb0	
return	xb1	
getstatic	xb2	y
putstatic	xb3	y
getfield	xb4	y
putfield	xb5	y
invokevirtual	xb6	y
invokespecial	xb7	y
invokestatic	xb8	y
invokeinterface	xb9	y
xxunsedxxx	xba	y
new	xbb	y
newarray	xbc	y
anewarray	xbd	y
arraylength	xbe	y

FIGURE 7C

athrow	xbf	y
checkcast	xco	y
instanceof	xc1	y
monitorenter	xc2	y
monitorexit	xc3	y
wide	xc4	y
multianewarray	xc5	y
ifnull	xc6	y
ifnonnull	xc7	y
goto_w	xc8	
jsr_w	xc9	
ldc_quick	xcb	y
ldc_w_quick	xcc	y
ldc2_w_quick	xcd	y
getfield_quick	xce	y
putfield_quick	xcf	y
getfield2_quick	xd0	y
putfield2_quick	xd1	y
getstatic_quick	xd2	y
putstatic_quick	xd3	y
gtestatic2_quick	xd4	y
putstatic2_quick	xd5	y
invokevirtual_quick	xd6	y
invokenonvirtual_quick	xd7	y
invokesuper_quick	xd8	y
invokestatic_quick	xd9	y
invokeinterface_quick	xda	y
invokevirtualobject_quick	xdb	y
new_quick	xdc	y
anewarray_quick	xde	y
multinewarray_quick	xdf	y
checkcast_quick	xe0	y
instanceof_quick	xe1	y
invokevirtual_quick_w	xe2	y
getfield_quick_w	xe3	y
putfield_quick_w	xe4	y
breakpoint	xca	y
impdep1	xfe	y
impdep2	xff	y

FIGURE 7 D